2017년 춘계학술발표대회 논문집 제24권 제1호(2017. 4)

에이전트 시스템 개발도구에 관한 연구

Dung T. Nguyen, 염상길, 추현승
성균관대학교 소프트웨어대학
e-mail : {ntdung, sanggil12, choo}@skku.edu

A switching-based delay optimal aggregation tree construction: An algorithm design

Dung T. Nguyen, Sanggil Yeom, Hyunseung Choo
College of Software, Sungkyunkwan University

요 약

Data convergecast is an indispensable task for any WSN applications. Typically, scheduling in the WSN consists of two phases: tree construction and scheduling. The optimal tree structure and scheduling for the network is proven NP-hard. This paper focuses on the delay optimality while constructing the data convergecast tree. The algorithm can take any tree as the input, and by performing the switches (i.e. a node changes its parent), the expected aggregation delay is potentially reduced. Note that while constructing the tree, only the in-tree collisions between the child nodes sending data to their common parent is considered.

1. Introduction

Wireless Sensor Networks (WSNs) are an essential part of the modern world technologies. One can easily find the presence of WSNs in both indoor and outdoor (monitoring, battlefield, forest fire alarm) applications. Nowadays, with the rise of size, large-scale WSNs are increasingly attracting researchers' effort.

Data convergecast, a particular type of data aggregation [1], where all the sensor nodes have to send its data to a sink node in an multi-hop manner. Sensor data are usually correlated, so that in-network aggregation can be applied. Each sensor node after getting others' data packets (i.e. one packet per node), can merge with its own data and generate a single data packet, and then send toward the sink. A data aggregation tree is usually built to guide the transmissions.

As the wireless nodes send data through a shared medium, collisions is an important factor that one should consider while designing a scheduling algorithm. One dominant approach is to use TDMA scheduling to alleviate the collisions. Typically, a TDMA scheduling method in the WSN consists of two phases: tree construction and scheduling. The tree constructed in the first phase will be used to guide the scheduling in the second phase. However, too strictly based on the pre-constructed structure may cause under-utilization of time slots.

Existing schemes usually build the Shortest Path Tree (SPT) or rely on the Connected Dominating Set (CDS) as the convergecast structure guiding the scheduling phase [2]. However, SPT and CDS may result in a large delay lower bound [3]. Let $G=(E,V)$ be the network. Malhotra et al. [2] for a particular tree, the data convergecast delay's lower bound is $\max\{|C_v|+h_v,\ \forall v \in V\}$, in which $|C_v|$ and h_v are the number of child nodes in the tree, and the depth of the node in the tree, respectively. They proposed a Balanced Shortest Path Tree (BSPT): minimizing the maximum child number of every node in each depth of the tree, so that the lower bound $\max\{|C_v|+h_v,\ \forall v \in V\}$ is hoped to be minimized. Although the effect of the depth of a tree on the convergecast delay is minimized in the SPT, it does not consider the link conflict, especially the region near the sink node. In [3], Pan et al. proposed the MLST that frees the tree construction from the SPT. They built the tree based on the Prim algorithm [] with the revised cost of the links. Their focus is also to minimize the $\max\{|C_v|+h_v,\ \forall v \in V\}$ of the whole tree.

So far, the tree construction process heavily relies on a predetermined heuristic algorithm. In this work, we propose a dynamic and robust algorithm that runs on top of any constructed tree, but can improve the expected convergecast delay without the consideration of all collisions. Only the conflicts between child nodes of a common parent are considered. The collision between nodes of different parents will be alleviated in the scheduling phase, which is also our future work.

In this paper, we design an enhancement algorithm to improve a given data convergecast structure. The structure of this paper is as follows. In Section 2 we provide some preliminaries and problem formulation. In Section 3, we illustrate our design in Section 4. Finally, Section 5 claims the novelty and suggests the possible directions in the future.

2. Preliminaries

A. *The Network Model*

A WSN consists of one sink node and a number of nodes having data across the deployment field. All the nodes are static, and with a limited transmission range. We assume that except the sink node, all the node are indentical in capacity or capability. For a covergecast (or aggregation equivalently), all the nodes have data to send. The data will follow the tree links to avoid redundant transmissions. Each node merges its received data and its own data, and then only produces a single packet toward the sink node.

B. Problem Formulation

At a time point, all the nodes have their data to send to the sink. To avoid collisions, the nodes use TDMA (Time Division Multiple Access) to send data to the sink. The problem we are interested in is how to build a data covergecast tree, and together with that, a schedule that offers good performance regarding data covergecast delay. Formally, data covergecast delay can be described as follows.

Given a wireless network represented by a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, in which $\boldsymbol{V}$ is a set of nodes, and $\boldsymbol{E}$ contains all the edges. There is one sink node $\boldsymbol{s} \in \boldsymbol{V}$. Let $\boldsymbol{S_i}$ be the set of scheduled senders sending data in time slot $\boldsymbol{i}$. A data covergecast schedule is defined as $\mathrm{S} = \bigcup_{\mathrm{i}}^{\mathrm{L}} \mathrm{S_i} = \mathrm{V} \setminus \{\mathrm{s}\}$, where $\boldsymbol{S_i} \cap \boldsymbol{S_j} = \emptyset, \forall \boldsymbol{i} \neq \boldsymbol{j}$; and $\boldsymbol{L}$ is the data covergecast delay. We have to find a tree structure and schedule that provides as small $\boldsymbol{L}$ as possible.

3. Proposed Design

A. The Ready Time

Ready time of a node is the *earliest* time when it gets all the data from the descendants (if any). Ready time of a node u is noted as rT_u. For any leaf node v, it is trivial that $rT_v = 0$. We can see that ready time of a sink node is also the data covergecast delay of the whole network.

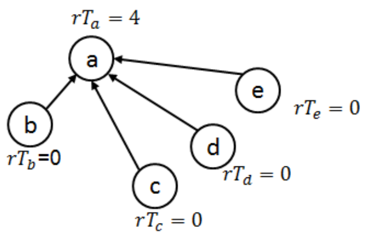

Figure 1 Example of ready time calculations

Theorem 1: Given the parent v with k children $v_1, v_2, \dots v_k$ whose ready times are sorted in a non-decreasing order: $r_1, r_2, \dots r_k$. Then $rT_v = \max\{r_i + 1 + k - i \mid 1 \leq i \leq k\}$

Proof:

Necessary: Let $s = \min(rT_v)$. Since $\{r_i\}$ is sorted in non-decreasing order, after r_i, there are $(k - i)$ nodes that need to be allocated time slots. Because any two nodes should take 2 different time slots; so at least $(k - i)$ time slots after $r_i + 1$ are needed. As a result, $s \geq (r_i + 1 + k - i),\ 1 \leq i \leq k$.

Sufficient: We just need to prove that if $s = \max\{r_i + 1 + k - i \mid 1 \leq i \leq k\}$, there is a valid time slots allocation for the nodes. Set $t_i' = s - k + i$, by the definition, we have $s - k + i \geq r_i + 1 + k - i - k + i = r_i + 1$. That is a valid assignment, hence the theorem is proved.

Considering the example in Figure 1, as given the parent a with four child nodes: b, c, d, e. By the Theorem 1, we can easily calculate the $rT_a = \max(0 + 1 + 4 - \mathrm{i} | 1 \leq i \leq 4) = \max(5 - i | 1 \leq i \leq 4) = 4$.

B. A Switch

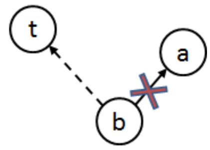

Figure 2 A switch

As in theorem 1, the ready time of a node depends some how on the children. Removing any child node will potentially result in a reduction in the parent's ready time. This can affect the ancestors of the parent on the way to the sink node. We define a unit switch as a simple operations that (a) a child node abandons its current parent, and (b) then the child node seeks for another parent candidate that least increases the overall delay of the network. In the best case, we got a reduction in step (a), and in step (b), there is no increment in the delay of the network caused by attaching the child node to a new parent.

C. Switching criteria

In order to get a better delay performance, the current tree structure should be changed by a number of switches. However, as the problem of optimally build a tree for the data convergecast is NP-hard, we cannot just use the brute-force approach. The algorithm should be able to identify a 'bad region' or 'bad node' that has more priority to be changed than others.

If we expect a reduction after one switch, the following observations should be considered:

- Removing a child node should reduce the parent's ready time (rT)
- Switching to a new parent may increase the delay at that parent. The algorithm should find a parent that the increment is as small as possible. This is to reduce the effect up to the sink node.

Based on those, we propose a priority function to determine a node whose one of its children should find another parent candidate:

$$f(a) = \alpha * \frac{hopCount_a}{\max hopCount} + \beta * \frac{indegree_a}{\max indegree}$$

In which $hopCount_a$ means the hop-count of node a to the sink node, via the tree links, $indegree_a$ means the number of children. A node with highest value will be selected. One of its child node will switch to a new parent. The algorithm is supposed to loop a number of times.

4. Conclusion

In this paper, we presented an idea of improving a given tree structure to support the data convergecast operation. Our main focus is the delay efficiency of the network. We derived a closed form formula to evaluate the network delay. The robust feature of our proposed scheme is that it can run on any starting tree structure, and if there is any chance to improve the delay performance, the algorithm might find out. In the future, we will implement the algorithm to check the effect of the algorithm, improve the switching criteria for better convergence, and reduce the complexity.

ACKNOWLEDGMENT

본 논문은 기초연구사업(NRF-2010-0020210)과 Grand ICT 연구센터지원사업 (IITP-2015-R6812-15-0001)의 연구결과로 수행되었음.

References

[1] Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal. "*Wireless sensor network survey*." Computer networks 52.12 (2008): 2292-2330.

[2] Malhotra, Baljeet, Ioanis Nikolaidis, and Mario A. Nascimento. "*Aggregation convergecast scheduling in wireless sensor networks*." Wireless Networks 17.2 (2011): 319-335.

[3] Pan, Cheng, and Hesheng Zhang. "*A time efficient aggregation convergecast scheduling algorithm for wireless sensor networks*." Wireless Networks 22.7 (2016): 2469-2483.